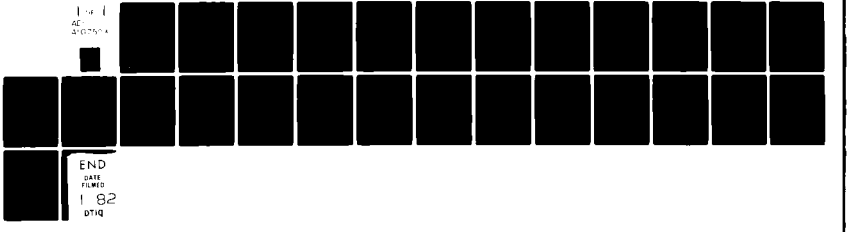
AD-A107 593 DREXEL UNIV PHILADELPHIA PA F/G 12/1
PLANNING IN A CONTINUOUS DOMAIN -- AN INTRODUCTION.(U)
NOV 81 R M SALTER N00014-80-C-0752
UNCLASSIFIED TR-1-81 NL



END
DATE FILMED
1 82
DTIC

LEVEL II

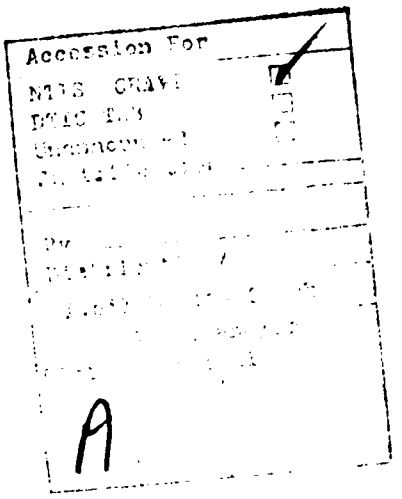
12

Technical Report #1-81
ONR Contract N00014-80-C-0752

AD A107593

Planning in a Continuous Domain -- An Introduction

Richard M. Salter
Drexel University



Planning in a Continuous Domain -- An Introduction

Richard M. Salter
Drexel University

ABSTRACT

This paper concerns the development of a model for planning over a domain of continuous actions. The envisioned world model consists of a set of piecewise defined functions of time, with critical points that are represented symbolically as instantaneous discrete events. The goal is represented as a point through which the world function must thread at some point in time. World functions are transformed by adding events at various points in time which transform the trajectories in the direction of the goal. As events are added, formulas relating the event times are added to the description of the world. The result is a partial order of events which represent a plan for "forcing" the world function to achieve the goal.

1. Introduction

The purpose of this report is to describe a new approach to the design of planning systems which are capable of reasoning about time. The model which is to be developed is based on ideas used in the CONCUR simulation system [Sal80], which is, in turn, based on ideas for continuous simulation first described in [He73]. The use of algebraic expressions to represent continuously varying parameters seems to lend itself quite naturally to simulation problems, but for the dual task of planning to achieve some specified goal, the perception of such expressions as representing infinitely many states of infinitesimal duration leads to an intractable search. In this paper we discuss a means for greatly limiting the number of alternatives in a continuous planning system through an approximation of the functions defining world model parameters, and by limiting the ways in which such functions may be altered. It is then possible to sketch a procedure for searching the state space which is analogous to that used in classical planning environments (e.g. [Ni80], Ch. 7 and 8). The solutions produced in this model contain both a plan (i.e. a set of actions) and a set of constraint equations and inequalities which fix the relationships between event times, usually including a partial order. This generalizes the idea of "non-linear planning" first discussed in [Sac75].

In the next section we briefly describe the proposed model. Section 3 contains an example of a planning problem which can be solved using this methodology. We conclude in section 4 with an analysis and prospectus of considerations for implementing these ideas.

2. Description of the Model

In the CONCUR system, production rules are used to define instantaneous events which act as boundaries to continuous processes [Sal81]. Thus, for example, the process of moving a robot from point A to point B is simulated using two events: the first starts the robot's motion by inserting an expression into its location slot, and the second stops the motion by replacing the expression by a constant. The "motion" itself is represented by the expression, which may be evaluated at any point in the intervening model time to give a location value for the robot. The function describing the robot's position over all time can be described as "piecewise defined" (and in this case piecewise continuous) since its value can be determined using a finite set of expressions and a finite partition of time intervals. This function is depicted in Figure 1.

In world modeling, we are interested in the time-dependent values of the various parameters that define the model, particularly in the way in which parameter

interactions affect these values. Given a rich enough vocabulary from which to construct expressions, it is possible to approximate the time functions of parameters that occur in many applications. This fact was used advantageously in the CONCUR system. It is also possible to assume that interactions in the model manifest their effects on a given parameter by altering the trajectory of that parameter, producing a point of change in its piecewise definition. We may therefore view the points at which the definition of a piecewise defined function changes as modeling the instantaneous events which act to alter the trajectory of that function. In the above example, two such events ("start" and "stop") can be attached to the change points in the graph. We shall limit ourselves to actions which result in these sorts of instantaneous events. The definition of an event therefore involves the specification of a new trajectory for certain world model parameters. The actual specification will turn out to be dependent on the values of parameters at the point in model time at which the event occurs.

The classical problem solving model is a set of discrete states and a set of (partial) state transformations. Both an initial state and a set of goal states are given, and the problem is to find a sequence of transformations which transform the initial state to some element of the goal set [Ba80]. In robot planning, the state description and goals are given by predicate calculus

wffs. In the STRIPS model, state transformations are described using precondition formulas and add and delete lists which perform a syntactic alteration of the state description [Ni80]. The blocks world state in Figure 2 can be described using a minimum of three conjuncts. Alternatively, we could represent all such states using three functions giving the position of the block (i.e. what object the particular block is on), as shown in Figure 2b. The advantage of this representation is that it can be extended to give a complete description of the location of the block over time. In Figure 3a we see a representation of this state as a function of time when no action is taken to change it. When a STRIPS-like plan is applied we obtain Figure 3b, a set of functions describing a world of abrupt changes, like a series of still photographs. Note that the three verticle lines at which these changes occur each represent an application of a STRIPS rule. The true picture is actually more like the one given in Figure 3c, since we know that blocks do not disappear from one location and instantly reappear in another. In this example, since only one diagonal appears in any vertical strip, there is no loss in the compressing of verticle strips that transforms Figure 3c to Figure 3b. On the other hand, if we were modeling a situation in which two arms were moving the blocks, so that two such diagonals could occur at the same time, it would be necessary to pay more attention to the position values between the STRIPS states due to the restrictions imposed on the intersection of position functions by the world being

modeled.

Motivated by this example, we define a world function to be a vector of piecewise defined functions of time. Each function can be defined using some conditional form (i.e. CASE or IF-THEN-ELSE) and an appropriate set of expressions, but even within a given structure there will clearly be definitions which are syntactically different but which define the same semantic function. This fact will not create any difficulty, since the syntactic transformations performed on these functions will preserve this equivalence.

A given world function provides a complete description of some possible world over an interval of time. We can interfere with this history at various points in time by altering the trajectories of some of the world function's components. This alteration transforms the world so that the resulting set of functions are identical to the original set up until the time of the event. For a given domain, we are provided with both the set of functions modeling the domain, and the set of possible events which transform world functions. In the next section, we shall see one example of how this representation can be used to solve a planning problem.

3. Example - "The Conveyor Belt Problem"

In order to illustrate the kind of application which can benefit from this representation we now present a simple example which cannot be modeled discretely. We shall refer to this as the "conveyor belt problem", as the model involves a block (B) moving along a conveyor belt at a constant speed. Ten horizontal units from the point at which the block first appears, and ten vertical units above the belt, is a robot arm which is capable of grasping the block. The arm may only move up and down, and does so at a constant speed. Figure 4a presents a picture of this world.

We can represent the state of the world using four real-valued (or two vector-valued) functions, LA and LB, giving the x and y coordinate positions for both the arm and the block as functions of time, respectively. In the initial world there is no action besides the movement of the block along the belt. In this world, the block appears at point (0,0) at some intial time, and moves along the belt and out of the scene. A picture of this initial world is given by the graphs in Figure 4b. We shall consider this entire scenario to be a single state, representing one possible world undergoing evolution in time. This notion of state is independent of individual points in time, and corresponds to the abstraction represented by the world function. We shall see that these sorts of states are the semantic counterparts of syntactic world functions.

If we state a goal, such as ∃t LB(t) = (10,10) (represented by the two crosses in the graph), we see that the initial world cannot satisfy this goal; in particular, the value of LBy is a constant 0. A solution, showing five events associated with the descent of the arm, the grabbing of the block, and the subsequent re-ascent of the arm, is given in Figure 4c. By adding these five events, we have transformed the world into one whose trajectory passes through the goal.

The solution given in Figure 4c is only one of infinitely many possible solutions, given the fact that the periods during which the arm may move have some degree of freedom. From the diagram, it is clear that the vertical strips in which the arm descends and ascends can be shifted either left or right to some extent. A complete solution must specify both the events to be executed (i.e. STARTDOWN; STOP; GRASP; STARTUP; STOP) and a set of constraints on the times at which these events are to occur. In this example, the time constraints will include a total order, but this need not be the case when the events produce non-interfering trajectories. The result is akin to Sacerdoti's non-linear plans [Sac75]. Another important point is that not every set of piecewise-defined graphs constitute a real solution to the problem--the block cannot "magically" float to the desired location. The solution must be developed with respect to a set of rules describing allowable transitions, and a set of axioms which must hold

between transition times.

We now propose one possible structure for solving this sort of planning problem. In it, events play the role of operators, and are designed to maintain the transition rules. A fixed set E of allowable events is given as part of the problem. Each event e in E is parameterized by the time, t, at which it is to occur. The event e is defined first by a set of preconditions which must hold near (but prior to) t in order for e(t) to take place. The action of e(t) is to impose a new trajectory on some of the components of the world function, whose specification may be based on the values of the world at time t. These restrictions permit the modeling of a well defined set of plausable transitions. It is important to note that these transitions do not map individual states forward in time, as is generally the case in discrete models. Indeed, states in this model constitute the entire scope of time for some world on some interval. Instead we are mapping between possible futures, making discrete transitions from one continuum to another, with transformations chosen from a continuum of possibilities.

In our example, there are four events, corresponding to the possible actions of the robot arm. These are labeled STARTDOWN, STARTUP, STOP and GRASP. We can describe a STRIPS-like procedure which employs a primitive pattern matcher to devise a plan and a set of constraining equations on event times. The times ti and tf correspond to the

bounds of the interval of interest. ti, the lower bound, is the time at which the block first comes into view (at (0,0)), and tf represents the time at which the goal is achieved. We will introduce indexed time variables as we need them for event times. The variable t0 is a bound variable used in Figure 5a to define the preconditions and trajectories of the four events. In this procedure we also differentiate between variables which are only matched to constant expressions (upper-case) and those which only match non-constant formulas (lower-case). This facilitates the search process by eliminating certain useless transitions, like stopping an arm which is not moving. Lower case variables are also used to denote the functions represented by the formulas to which they are bound. As in CONCUR, we also preceed *each non-constant* formula used in the description of a world function with an asterisk, indicating that evaluation is required. Goals and subgoals are specified along with an interval on which events may be added. Since an event only affects the value of the world function following the time of its occurrance, specifying this interval serves to protect subgoals which have already been achieved. The equations relating event times result from the unification achieved during the matching process, and are an integral part of the solution. These equations are in fact subgoals which define classes of solutions. The procedure for achieving the goal LB(tf) = (10,10) is given in Figure 5.

Some further remarks on this example are in order. Figure 5 only identifies a single path through the search tree. We have surpressed all of the non-fruitful paths for the sake of brevity. Some of these paths quite obviously lead to a dead end: for example, attempting to apply Startdown when the arm is in its lowest position in order to achieve the subgoal LA(t5) = (10, z), when z(t4) = 10 and t5 < t4 produces the equation *(-sp1 x (t4 - t5) + 0) = 10, which cannot be solved. Insolvability of resulting equations is one criterion (although not always useful) for ruling out branches of the tree. It is usually the case, however, that the equations contain several time variables, and are therefore underspecified. It is therefore necessary to develop a precise metric for discriminating between branches of the search tree. This metric should combine some aspects of the problem domain (heuristics) with some analysis of the equations produced by that branch.

We have also found that the constraints on the world, in this case describing the limitations on the movement of the arm and block, were easily maintained by the events. This may not generally be the case, especially in situations involving many more world function components. The problem of maintaining the validity of certain axioms over time intervals may be impossible to solve unless we limit the restrictions imposed on trajectories to those which can be verified through some reasonable procedure like, for instance, equation solving. We do not believe that this

limitation greatly restricts the domains which can be modeled, and will ultimately simplify any program implementing these techniques.

Another point concerns the trajectories specified by the four event scenarios. Three of the four described trajectories based directly on parameterized expressions or bindings obtained during the matching process. The fourth scenario, GRASP, imposed a trajectory which defined one world function component in terms of another. The algorithm handled this situation by substituting a new goal involving the assigned component. It is clear that such trajectories must be permitted, so long as they do not result in a recursive definition. In general, it will be necessary to handle trajectory assignments which involve more complex expressions involving other components, and a methodology for handling this situation is required.

Finally, we note that the use of pattern matching was critical in obtaining the crucial time relationships, but the patterns were not especially complex. The simple syntactic structure of events and goals does not seem to require more than a simple matching mechanism, but any such mechanism must be capable of deducing the constraint equations which are produced through the matches.

4. Analysis and Conclusion

This informal algorithm for solving the conveyor belt problem shows how conventional planning techniques can be used with this model to reason about continuous time. From the remarks given in the last section, it is clear that a more precise analysis of the model is required to satisfy some of the difficulties which will arise when we attempt to apply it to more complex situations. One such precise mathematical description is given in the Appendix. In this description, we give definitions for the syntax and semantics of world functions, events, processes (sequences of events) and plans (a process together with a set of time constraints), and define the planning problem in terms of these definitions. We hope that this mathematical model will serve as a point of reference for a precise development of a generalized procedure.

In order to achieve a realistic implementation it may be necessary to make some simplifying assumptions. These may include the following: 1) Allow only simple predicates (e.g. "=", "<", ">", etc.) for goals and subgoals, and specify intervals on which events may be added; 2) Permit goals and subgoals only of the form $\exists t$ (Fi1(t) = f1) & ... & (Fik(t) = fk), specifying the values of certain parameters at a time slice; 3) Apply similar restrictions to event preconditions; 4) Limit axioms to ones which can be verified by equation solving (this actually has the added benefit of identifying points which are candidates for events); and 5) Limit trajectories to monotone functions

(making it easy to identify constant trajectories).

In solving the planning problem, we must be able to determine for a goal G the choice of an event e and the placement of e at some point in time. For each choice and placement, a set of subgoals is produced consisting of event preconditions and the regressed goal G' (for any predicate P and any state transformation T, the predicate P' is the regression of P through T if for any state s satisfying P' we have that T(s) satisfies P, and P' is the weakest computable predicate with this property, see [Wa77]). The computation of G' will produce the equations which constrain the event times, as well as possibly producing alternate goals (as in the case of GRASP above). The determination of which event to choose and where to place it will depend on an analysis of the computed subgoal. We certainly require that progress be made on achieving the goal, so that only events whose trajectories affect G should be considered (otherwise G' will be identical to G). Other criteria include solvability of time constraints, "means-ends" analysis [Ne63] of subgoal and current world, and possible violation of axioms. We can further constrain the search by ordering goal conjuncts so that the placement of events can be made according to whether or not higher order constraints are immediately satisfied. Further restrictions of event times can be made according to whether or not certain axioms are violated.

Finally, it may be beneficial to recognize the fact that certain events are elements of larger units of behavior (e.g. "start" and "stop" are a part of the process "move"), making a higher level of motivation possible in the design of the plan. This would involve some hierarchical planning structure [Sac77] with layers composed of sequences of processes, but which ultimately results in some set of instantaneous events. Events are the ultimate units of any hierarchial structure.

In this paper we have introduced a method for modeling domains of continuous time and have introduced an approach to the design of algorithms for planning in such domains. The key points are the fact that a given state involves a complete history over some time interval, rather than specifying parameter values at some point in time, and that state transformations, or events, are instantaneous and transform states according to trajectories that eminate from the time of the event. We believe that further development along the lines described above will produce a precise methodology for an implementation of a planning system using this model.

Acknowledgement

The author wishes to thank Stanley J. Rosenschein for many helpful discussions during the author's visit to SRI International. This report describes research undertaken at the Artificial Intelligence Center at SRI International during the summer of 1981, supported by the Office of Naval Research under contract N00014-80-C-0752.

References

[Ba80] R. B. Banerji, "Artificial Intelligence -- A Theoretical Approach", North Holland, 1980.

[He73] G. Hendrix, "Modeling simultaneous actions and continuous processes", Artificial Intelligence, vol. 4 (1973), pp. 145-180.

[Ne63] A. Newell and H. A. Simon, "GPS, a program that simulates human thought", in Computers and Thought (Feigenbaum and Feldman, eds.), McGraw-Hill, 1963, pp. 279-293.

[Ni80] N. J. Nilsson, "Principles of Artificial Intelligence", Tioga Publishing Co., 1980.

[Sac75] E. D. Sacerdoti, "The non-linear nature of plans", Proc. 4th IJCAI (1975), pp. 206-214.

[Sac77] _____, "A Structure for Plans and Behavior", Am. Elsivier, 1977.

[Sal81] R. M. Salter, "Time and production systems", in progress.

[Sal80] _____, T. Brennan, and D. P. Friedman, "CONCUR: A language for continuous, concurrent processes", Computer Languages, vol. 5 (1980), pp. 163-189.

[Wa77] R. Waldinger, "Achieving several goals simultaneously", Machine Intelligence, vol. 8 (1977), pp. 94-136.

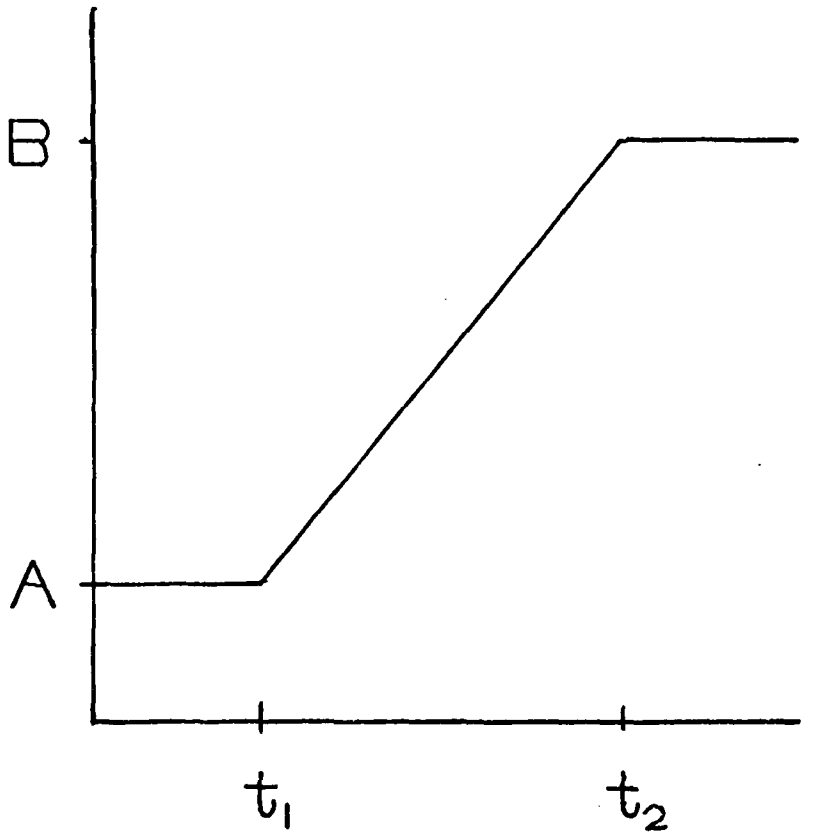
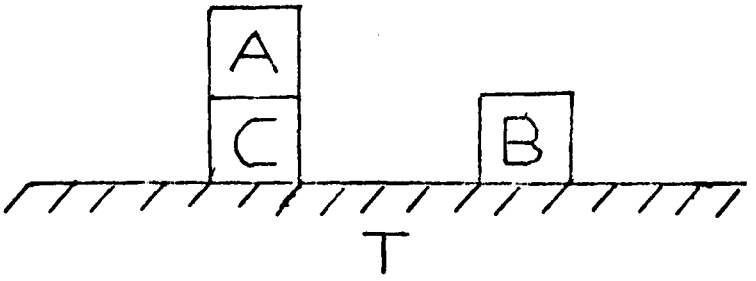


Figure 1: Graph of Robot's Motion

S:



a)

$$ON(A,C) \wedge ON(B,T) \wedge ON(C,T)$$

b)

$$ON\text{-}A(S) = C$$
$$ON\text{-}B(S) = T$$
$$ON\text{-}C(S) = T$$

Figure 2: Representing a Blocks World State

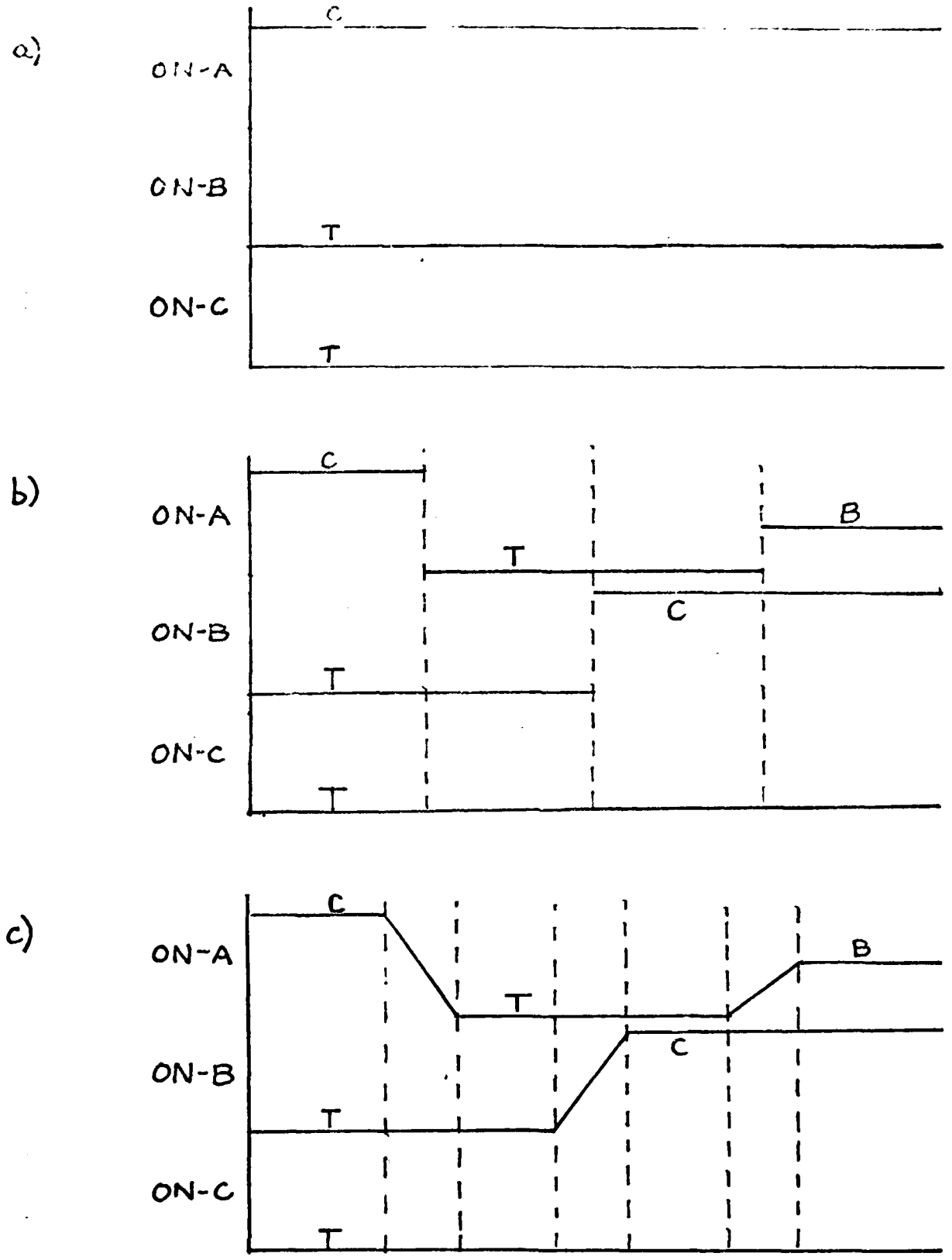


Figure 3: Blocks World Over Time

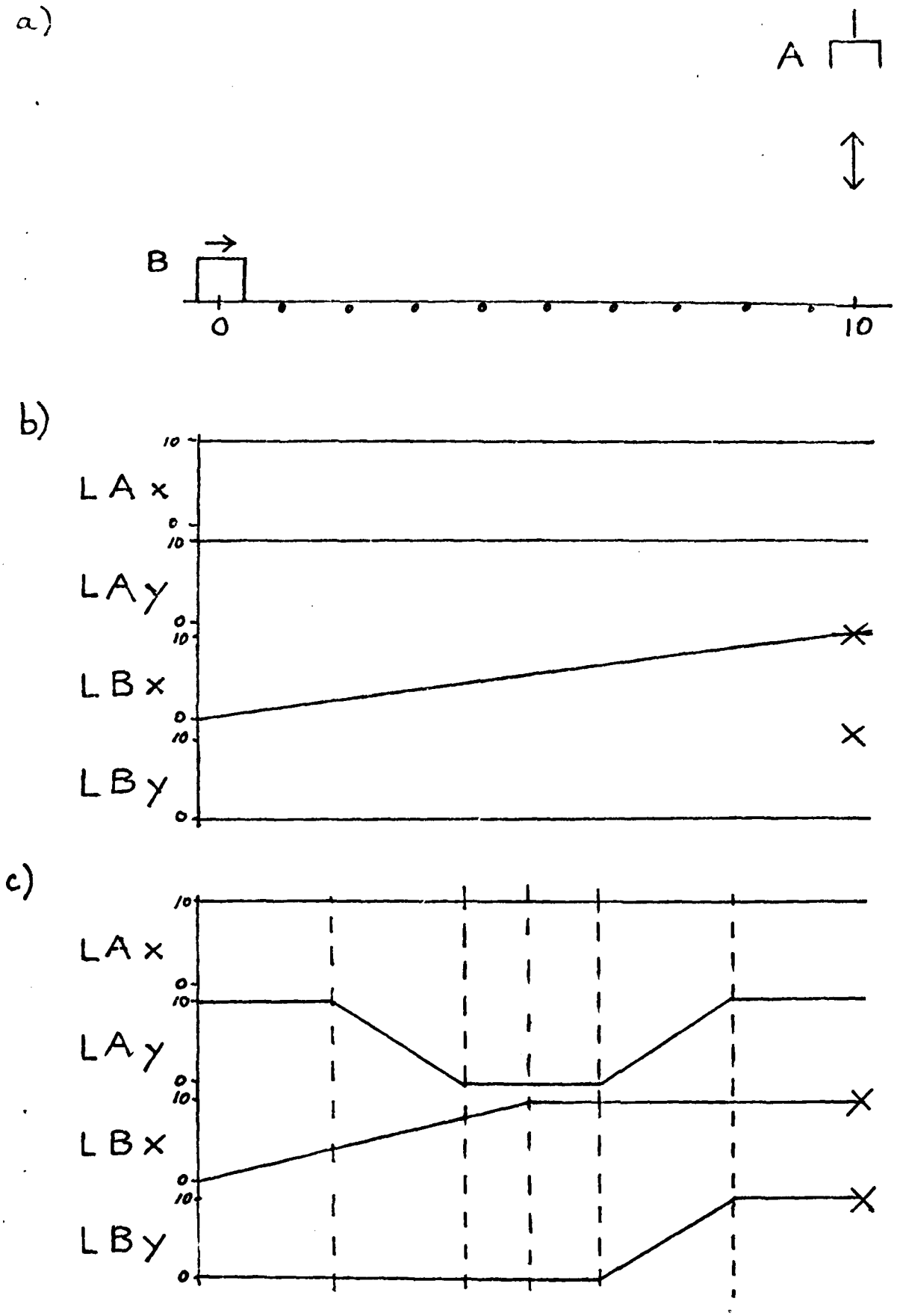


Figure 4: Conveyor Belt Problem

Figure 5 -- Conveyor belt problem solution

a) EVENTS:

```
        GRASP(t0)
pre:  LA(t0) = (10, X)
      LB(t0) = (10, X)
tra:  LB(t) = LA(t)

        STARTUP(t0)
pre:  LA(t0) = (10, Y)
tra:  LA(t) =
        (10, *(sp1 x (t- t0) + Y))

        STARTDOWN(t0)
pre:  LA(t0) = (10, Y)
tra:  LA(t) =
        (10, *(-sp1 x (t - t0) + Y))

        STOP(t0)
pre:  LA(t0) = (10, z(t0))
tra:  LA(t) = (10, z(t0))
```

b) GOAL: LB(tf) = (10, 10) on [ti, tf]

AXIOMS: for all t:

```
LAx(t) = 10
0 <= LAy(t) <= 10
LBy(t) <> 0 ==> LA(t) = LB(t)
```

INITIAL WORLD: F0

```
LA(t) = (10, 10)                          [ti, tf]
LB(t) = (*(sp2 x (t - ti)), 0)            [ti, tf]
```

1 LB(tf) = (10, 10)
matches GRASP(t1) {tf/t, (10, 10)/LA(tf)}
s.s.: LA(tf) = (10, 10) [t1, tf] (6)
LA(t1) = (10, X) [ti, t1] (3)
LB(t1) = (10, X) [ti, t1] (2)
t.c.: t1 <= tf

2 LB(t1) = (10, X)
matches F0 {t1/t, 10/*(sp2 x (t1 - ti)), X/0}
t.c.: sp2 x (t1 - ti) = 10

3 LA(t1) = (10, X)
matches STOP(t2) {t1/t, X/z(t2)}
s.s.: LA(t2) = (10, z(t2)) [ti, t2] (4)
t.c.: t2 <= t1
z(t2) = X

4 LA(t2) = (10, z)
matches STARTDOWN(t3) {t2/t, z(t2)/*(-sp1 x (t2 - t3) + Y)}
s.s.: LA(t3) = (10, Y) [ti, t3] (5)
t.c.: t3 <= t2
-sp1 x (t2 - t3) + Y = z(t2)

5 LA(t3) = (10, Y)
matches F0 {t3/t, 10/Y}

Apply STARTDOWN(t3); STOP(t2); GRASP(t1); to transform F0 into F1:

```
        LA(t) = (10, 10)                              [ti, t3]
                (10, *(-sp1 x (t - t3) + 10))         (t3, t2]
                (10, 0)                               (t2, t1]
        LB(t) = (*(sp2 x (t - ti)), 0)                [ti, t1]
                LA(t)                                 [t1, tf]

                t3 <= t2 <= t1 <= tf
                sp2 x (t1 - ti) = 10
                -sp1 x (t2 - t3) + 10 = 0
```

6 LA(tf) = (10, 10)
matches STOP(t4) {tf/t, 10/z(t4)}
s.s.: LA(t4) = (10, z(t4)) [t1, t4] (7)
t.c.: t1 <= t4 <= tf
z(t4) = 10

7 LA(t4) = (10, z(t4))
matches STARTUP(t5) {t4/t, z(t4)/*(sp1 x (t4 - t5) + Y)}
s.s.: LA(t5) = (10, Y) [t1, t5] (8)
t.c.: t1 <= t5 <= t4
sp1 x (t4 - t5) + Y = z(t4)

8 LA(t5) = (10, Y)
matches F1 {t5/t, Y/0}

Apply STARTUP(t5); STOP(t4) to transform F1 into F2:

```
        LA(t) = (10, 10)                              [ti, t3]
                (10, *(-sp1 x (t - t3) + 10))         (t3, t2]
                (10, 0)                               (t2, t5]
                (10, *(sp1 x (t - t5)))               (t5, t4]
                (10, 10)                              (t4, tf]
        LB(t) = (*(sp2 x (t - ti)), 0)                [ti, t1]
                LA(t)                                 [t1, tf]

                t3 <= t2 <= t1 <= t5 <= t4 <= tf
                sp2 x (t1 - ti) = 10
                -sp1 x (t2 - t3) + 10 = 0
                sp1 x (t4 - t5) = 10
```

Plan: STARTDOWN(t3); STOP(t2); GRASP(t1); STARTUP(t5); STOP(t4)

Note: s.s. indicates logical subgoals; t.c. indicates time constraints (subgoals satisfied through equation solving). Numbers following subgoals point to section handling that subgoal. Match substitutions appear in brackets.

APPENDIX -- Formal Description

I World Function: F ∈ W on T = [t1, tf]

Syntax:

F = ((I1,...In), {fi,j; i=1,..m, j=1,..n});

(I1,...In) is a partition of left open intervals on T.

For all i,j, fij is an element of EXPi, a set of expressions which may include the symbols F1,...Fm.

Fi is a label applied to the ith row {fi,j; j=1,...n}.

Semantics:

For each i, there is a domain Di and an evaluation function

EVALi : EXPi --> T --> Di

If U = T --> D1 x ... x Dm, then {EVALi} induces

EVAL : W --> U

defined by

EVAL(F,t)|Di = EVALi(fi,j, t), where t ∈ Ij

II Event: e ∈ E

Syntax:

e = (P(t0), G).

P is a wff in an appropriate first order language.

G = (g1,...gn), gi ∈ EXPi(t0) U {NO-OP}.

Semantics:

DO : E --> T --> W --> W

F' = DO(e, t, F) is defined as follows:

If t is in Ik, define Ik1 = (tk-1, t]; Ik2 = (t, tk].

F' has partition I1,...Ik1, Ik2,...In.

If P(t) is false, then F' is undefined, otherwise for each i, if gi = NO-OP then Fi' = Fi, else

$$F_i' = (f_{i1},\ldots f_{ik}, \underbrace{g_i(t),\ldots g_i(t)}_{n-k+1}).$$

III Process of length r: p ∈ Pr

Syntax:

p = (e1,...er); ei events.

Semantics:

DOPr : Pr --> Tr --> W --> W,
where Tr = {(t1,...tr); t1 < t2 < ... < tr}, defined recursively:

DOP0 = Id
DOPr((e1,...er), (t1,...tr), F)
= DOPr-1((e2,...er), (t2,...tr), DO(e, t1, F)).

IV Plan of length r for F: q ∈ Qr(F)

Syntax:

q = (pr; C); pr ∈ Pr

C is a set of constraints on the symbols

{ti, t1, ... tn-1, tf, t1', t2', ... tr'}

Semantics:

DOQr : Qr --> Tr --> W, defined by

DOQr(q, (t1',...,tr')) is undefined if, when ti, t1, ... tf are interpreted as the boundaries of the partition of F, any of the constraints in C become false; else

DOQr(q, (t1',...,tr'))
= DOPr(q, (t1',...,tr'), F).

V Planning problem: Given F0 ∈ W and a finite set of axioms

A ⊆ FOL({=, <, ...}, {F1,...Fm} ∪ {+, -, x, ...}),

find a plan q ∈ Qr(F0), for some r, such that DOQ(q, (t1',...,tr')) satisfies A for all t1',...,tr' satisfying C (where, as above, ti,...,tf are interpreted as the boundaries of the partition of F).

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: Drexel Technical Report 1-81	2. GOVT ACCESSION NO: AD-A107 593
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): 6 Planning in a Continuous Domain - An Introduction	5. TYPE OF REPORT & PERIOD COVERED: (9) Technical Report, 1 Nov 1981
6. PERFORMING ORG. REPORT NUMBER: ONR NR 049-480	
7. AUTHOR(S): (10) Richard M. Salter	8. CONTRACT OR GRANT NUMBER(S): (15) N00014-80-C-0752
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Drexel University, Philadelphia, PA 19104	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 11 1 84
11. CONTROLLING OFFICE NAME AND ADDRESS: ONR Code 411 IS, Arlington, VA (14) TR-1-82	12. REPORT DATE: 11/81
	13. NUMBER OF PAGES: 24 (12) 27
14. MONITORING AGENCY NAME & ADDRESS(If different from Controlling Office)	15. SECURITY CLASS. (of this report): U
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Robot Modeling, Planning Systems, Continuous Processes

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper concerns the development of a model over a domain of continuous actions. The envisioned world model consists of piecewise defined functions whose trajectories are transformed by events. A partial ordering of events and the forcing of a goal are obtained by threading these world functions through various points in time.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

40 723

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)


DA
FILM